Privater Masterstudiengang
Programmierung von Videospielen

tech technologische universität

tech technologische universität

Privater Masterstudiengang
Programmierung von Videospielen

- Modalität: **online**
- Dauer: **12 Monate**
- Qualifizierung: **TECH Technologische Universität**
- Aufwand: **16 Std./Woche**
- Zeitplan: **in Ihrem eigenen Tempo**
- Prüfungen: **online**

Internetzugang: www.techtitute.com/de/informatik/masterstudiengang/masterstudiengang-programmierung-videospielen

Index

01

Präsentation

Der größte Reiz eines Videospiels liegt in seinen visuellen Aspekten wie Grafik und Design. Ohne die Programmierung könnten diese Aspekte jedoch nicht hervorstechen. Die Programmierung ist der Schlüssel zu jedem Videospiel, denn sie bestimmt die Spielbarkeit oder die Art und Weise, wie die Grafik mit dem Spieler interagiert. Ohne eine gute Programmierung wäre jedes Spiel zum Scheitern verurteilt, da es viele *Bugs* hätte und keine angenehme Erfahrung wäre. Die Unternehmen sind sich dessen bewusst und benötigen daher hochqualifizierte Entwickler. Dieser Studiengang ist die Antwort auf diese Nachfrage, denn er bereitet die Studenten darauf vor, alle Herausforderungen der Branche zu meistern, und eröffnet ihnen zahlreiche berufliche Möglichkeiten.

Die Unternehmen wissen, dass der Schlüssel zum Erfolg eines Videospiels in der Programmierung liegt. Spezialisieren Sie sich und werden Sie der begehrteste Entwickler in Ihrer Umgebung”

Hinter jedem großartigen Videospiel steht ein riesiges Team von Fachleuten, die sich auf jeden einzelnen Arbeitsbereich spezialisiert haben und versuchen, ihr Unternehmen zum Erfolg zu führen. Für die Fans sind in der Regel die Bereiche am auffälligsten, die sie direkt wahrnehmen können, wie z. B. die Grafik oder die Bereiche, die mit der Steuerung der Charaktere, der Mechanik oder der Interaktion mit Objekten zu tun haben.

Damit all diese Elemente funktionieren und richtig integriert werden können, gibt es jedoch eine wesentliche Aufgabe, die normalerweise vernachlässigt wird: die Programmierung. Die Entwicklung eines Videospiels durchläuft verschiedene Phasen und umfasst verschiedene Abteilungen, aber die Programmierung ist diejenige, die dem Ganzen einen Sinn gibt und das Grundgerüst bildet, auf dem die anderen Bereiche aufbauen.

Aus diesem Grund widmen die Unternehmen der Branche diesem Bereich so viel Aufmerksamkeit, denn sie wissen, dass eine korrekte und effiziente Entwicklung ihrer Videospiele den Fortschritt des Projekts erleichtert und das Auftreten von Fehlern und *Bugs* verhindert. Aus diesem Grund suchen sie die besten Programmierer, die auf diesem Gebiet spezialisiert sind.

Es ist jedoch nicht einfach, echte Spezialisten auf diesem Gebiet zu finden. Der Private Masterstudiengang in Programmierung von Videospielen ist die Antwort auf diese Nachfrage. Er macht die Studenten zu Experten in der Entwicklung von Videospielen, die dank der in diesem Studiengang erworbenen Fähigkeiten und Fertigkeiten leichtin der Branche Fuß fassen können und große Karrierechancen haben.

Dieser **Privater Masterstudiengang in Programmierung von Videospielen** enthält das vollständigste und aktuellste Programm auf dem Markt. Die hervorstechendsten Merkmale sind:

- Die Entwicklung von Fallstudien, die von Experten für Programmierung und Entwicklung von Videospielen präsentiert werden
- Der anschauliche, schematische und äußerst praxisnahe Inhalt vermittelt alle für die berufliche Praxis unverzichtbaren wissenschaftlichen und praktischen Informationen
- Er enthält praktische Übungen, in denen der Selbstbewertungsprozess durchgeführt werden kann, um das Lernen zu verbessern
- Sein besonderer Schwerpunkt liegt auf innovativen Methoden
- Theoretische Vorträge, Fragen an den Experten, Diskussionsforen zu kontroversen Themen und individuelle Reflexionsarbeit
- Die Verfügbarkeit des Zugangs zu Inhalten von jedem festen oder tragbaren Gerät mit Internetanschluss

Entwickeln Sie mit diesem privaten Masterstudiengang alle Arten von Videospielen in den besten Unternehmen der Welt”

Die Programmierung wird bei der Entwicklung von Videospielen immer wichtiger. Werden Sie mit diesem Abschluss ein unverzichtbarer Teil der Branche”

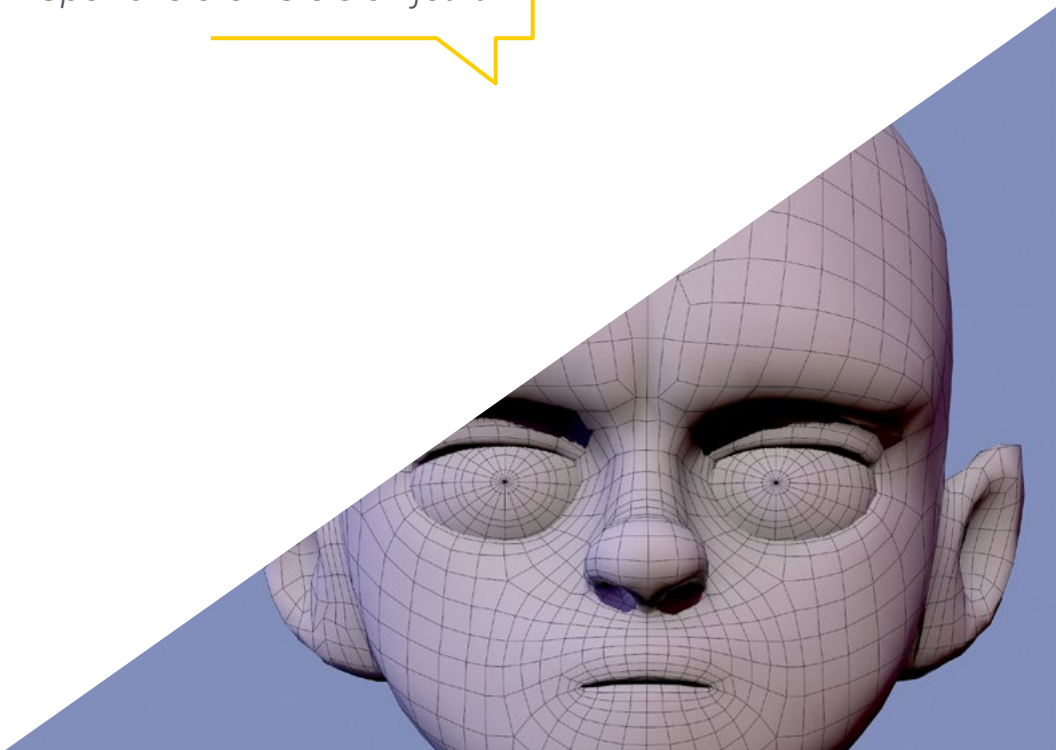
Spiele sind Ihre Leidenschaft und Sie wollen ein großer Entwickler werden. Warten Sie nicht länger und schreiben Sie sich für diesen privaten Masterstudiengang ein.

Die besten Unternehmen der Branche warten auf Sie. Spezialisieren Sie sich jetzt.

Zu den Dozenten des Programms gehören Fachleute aus der Branche, die ihre Erfahrungen aus ihrer Arbeit in diese Weiterbildung einbringen, sowie anerkannte Spezialisten aus führenden Unternehmen und renommierten Universitäten.

Die multimedialen Inhalte, die mit der neuesten Bildungstechnologie entwickelt wurden, werden der Fachkraft ein situiertes und kontextbezogenes Lernen ermöglichen, d. h. eine simulierte Umgebung, die eine immersive Fortbildung bietet, die auf die Ausführung von realen Situationen ausgerichtet ist.

Das Konzept dieses Programms konzentriert sich auf problemorientiertes Lernen, bei dem die Fachkraft versuchen muss, die verschiedenen Situationen aus der beruflichen Praxis zu lösen, die während des gesamten Studiengangs gestellt werden. Zu diesem Zweck wird sie von einem innovativen interaktiven Videosystem unterstützt, das von renommierten Experten entwickelt wurde.



02

Ziele

Das Hauptziel dieses privaten Masterstudiengangs ist es, Studenten zu großartigen Videospielentwicklern zu machen. Diese Branche expandiert und braucht immer mehr Programmierer und Spezialisten mit einer hochqualifizierten Weiterbildung. Daher ist dieser Abschluss perfekt für großartige Karrierechancen in einigen der renommiertesten Unternehmen der Welt. Dieser Studiengang bietet den Studenten also alle notwendigen Fähigkeiten, um in diesem Sektor zu gefragten Experten zu werden und einen bedeutenden und unmittelbaren Aufstieg in ihrer Karriere zu erreichen.

Mit diesem Privaten Masterstudiengang in Programmierung von Videospielen sind alle Ihre Träume in greifbare Nähe gerückt”

Allgemeine Ziele

- Erlernen der verschiedenen Programmiersprachen und Methoden, die bei Videospielen angewandt werden
- Vertiefen in den Produktionsprozess von Videospielen und die Integration der Programmierung in diese Phasen
- Erlernen der Grundlagen des Videospiel-Designs und der theoretischen Kenntnisse, die ein Videospiel-Designer kennen muss
- Beherrschen der grundlegenden Programmiersprachen, die in Videospielen verwendet werden
- Anwenden von Wissen über *Software Engineering* und spezialisierte Programmierung auf Videospiele
- Verstehen, welche Rolle die Programmierung bei der Entwicklung eines Videospiels spielt
- Kennenlernen der verschiedenen existierenden Konsolen und Plattformen
- Entwickeln von Web- und Multiplayer-Videospielen

Wenn Sie dieses Studium abschließen, werden Sie der beste Videospielentwickler in Ihrer Umgebung sein"

Spezifische Ziele

Modul 1. Grundlagen der Programmierung

- Verstehen der grundlegenden Struktur eines Computers, der Software und der allgemeinen Programmiersprachen
- Analysieren der wesentlichen Elemente eines Computerprogramms, wie z. B. die verschiedenen Datentypen, Operatoren, Ausdrücke, Anweisungen, E/A und Steueranweisungen
- Interpretieren von Algorithmen, die die notwendige Grundlage für die Entwicklung von Computerprogrammen sind

Modul 2. Datenstruktur und Algorithmen

- Kennenlernen der wichtigsten Algorithmus-Design-Strategien sowie der verschiedenen Methoden und Maßnahmen zur Berechnung von Algorithmen
- Unterscheiden der Funktionsweise von Algorithmen, ihrer Strategie und Beispiele für ihren Einsatz bei den wichtigsten bekannten Problemen
- Verstehen der *Backtracking*-Technik und ihrer wichtigsten Anwendungen

Modul 3. Objektorientierte Programmierung

- Erlernen der verschiedenen *Design Patterns* für objektorientierte Probleme
- Verstehen der Bedeutung von Dokumentation und Testen in der Softwareentwicklung
- Umgehen mit der Verwendung von *Threading* und Synchronisierung sowie die Lösung gängiger Probleme bei der gleichzeitigen Programmierung

Modul 4. Spielkonsolen und Geräte

- Verstehen der grundlegenden Funktionsweise der wichtigsten Ein- und Ausgabeperipheriegeräte
- Verstehen der wichtigsten Auswirkungen des Designs der verschiedenen Plattformen
- Studieren der Struktur, Organisation, Funktionsweise und Verbindung von Geräten und Systemen
- Verstehen der Rolle von Betriebssystemen und Entwicklungskits für mobile Geräte und Videospielplattformen

Modul 5. Softwaretechnik

- Unterscheiden der Grundlagen der Softwaretechnik sowie des Softwareprozesses und der verschiedenen Softwareentwicklungsmodelle einschließlich agiler Technologien
- Erkennen des Anforderungsmanagements, seiner Entwicklung, Ausarbeitung, Verhandlung und Validierung, um die wichtigsten Standards in Bezug auf Softwarequalität und Projektmanagement zu verstehen

Modul 6. Videospiel-Engines

- Entdecken der Funktionsweise und Architektur einer Videospiel-Engine
- Verstehen der grundlegenden Eigenschaften bestehender Videospiel-Engines
- Programmieren von Anwendungen, die korrekt und effizient auf Videospiel-Engines angewendet werden
- Auswählen des am besten geeigneten Paradigmas und der Programmiersprachen zur Programmierung von Anwendungen für *Video Game Engines*

Modul 7. Intelligente Systeme

- Erstellen der Konzepte der Agententheorie und der Agentenarchitektur und ihrer Argumentationsprozesse
- Verstehen der Theorie und Praxis hinter den Konzepten von Information und Wissen sowie der verschiedenen Arten der Wissensrepräsentation
- Verstehen der Funktionsweise von semantischen Reasonern, wissensbasierten Systemen und Expertensystemen

Modul 8. Echtzeit-Programmierung

- Analysieren der wichtigsten Merkmale einer Echtzeit-Programmiersprache, die sie von einer traditionellen Programmiersprache unterscheiden
- Verstehen der grundlegenden Konzepte von Computersystemen
- Erwerben der Fähigkeit, die wichtigsten Grundlagen und Techniken der Echtzeit-Programmierung anzuwenden

Modul 9. Design und Entwicklung von Webspielen

- In der Lage sein, Spiele und interaktive Webanwendungen mit der entsprechenden Dokumentation zu entwerfen
- Bewerten der Hauptmerkmale von Spielen und interaktiven Webanwendungen, um professionell und korrekt zu kommunizieren

Modul 10. Multiplayer-Netzwerke und -Systeme

- Beschreiben der Architektur des Transmission Control Protocol/Internet Protocol (TCP/IP) und der grundlegenden Funktionsweise von drahtlosen Netzwerken
- Analysieren der Sicherheit in Bezug auf Videospiele
- Erwerben der Fähigkeit, Multiplayer-Online-Spiele zu entwickeln

03

Kompetenzen

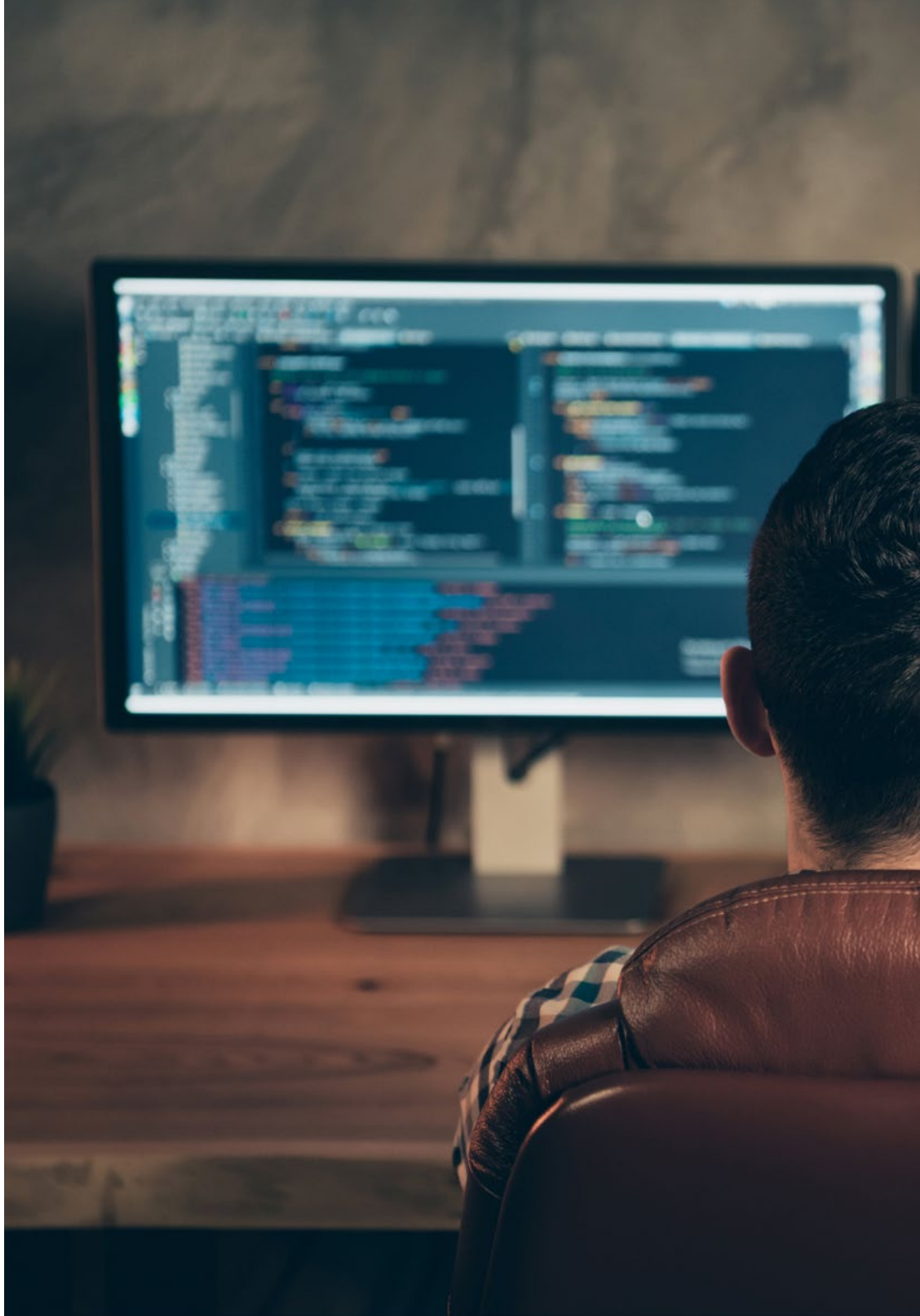
Dieser Private Masterstudiengang in Programmierung von Videospielen macht die Studenten dank der Fähigkeiten und Fertigkeiten, die er ihnen vermittelt, zu echten Spezialisten für die Entwicklung dieser Art von audiovisuellen Werken. Dank dieses ausgezeichneten Programms verfügen die Studenten über ein professionelles Instrumentarium, das sie in die Lage versetzt, alle mit der Programmierung von Videospielen verbundenen Herausforderungen zu meistern.

Sie werden alle Aspekte der Videospielentwicklung beherrschen"

Allgemeine Kompetenzen

- Entwerfen aller Phasen eines Videospiels, von der ersten Idee bis zur endgültigen Veröffentlichung
- Sich auf die Programmierung von Videospielen spezialisieren
- Vertiefen in alle Teile der Entwicklung, von der anfänglichen Architektur über die Programmierung des Spielercharakters bis hin zu allen am Spielprozess beteiligten Elementen
- Erhalten einer Gesamtvision des Projekts, um Lösungen für die verschiedenen Probleme und Herausforderungen zu finden, die bei der Entwicklung eines Videospiels auftreten

Erreichen Sie dank dieses privaten Masterstudiengangs hervorragende Leistungen als Videospielprogrammierer"



Spezifische Kompetenzen

- Kennen der Software, die Sie als professioneller Videospielentwickler benötigen
- Verstehen der Spielerfahrung und wissen, wie man die Spielbarkeit eines Videospiels analysiert
- Verstehen aller theoretischen und praktischen Vorgehensweisen bei der Programmierung von Videospielen
- Beherrschen der nützlichsten Programmiersprachen für die Welt der Videospiele
- Integrieren der erlernten Programmierung in verschiedene Arten von Konsolen und Plattformen
- Programmieren von Web- und Multiplayer-Videospielen
- Verinnerlichen des Konzepts der Videospiel-Engine, um korrekt programmieren zu können
- Anwenden von Kenntnissen der Softwaretechnik auf die Programmierung von Videospielen

04

Struktur und Inhalt

Die Inhalte dieses Privaten Masterstudiengangs in Programmierung von Videospielen wurden von einem Team aus führenden Experten auf diesem Gebiet, die den aktuellen Stand der Branche genau kennen, sorgfältig ausgearbeitet. Dieses Programm ermöglicht es den Studenten, sich alle notwendigen Kenntnisse anzueignen, um den Anforderungen der Unternehmen der Branche gerecht zu werden, da sie speziell auf die komplexen und sich ständig weiterentwickelnden Eigenheiten und Besonderheiten der Branche vorbereitet werden.

Diese Inhalte werden Sie zu einem großen Experten in der Programmierung von Videospielen machen"

Modul 1. Grundlagen der Programmierung

1.1. Einführung in die Programmierung
- 1.1.1. Grundlegende Struktur eines Computers
- 1.1.2. Software
- 1.1.3. Programmiersprachen
- 1.1.4. Lebenszyklus einer Softwareanwendung

1.2. Algorithmusentwurf
- 1.2.1. Lösung von Problemen
- 1.2.2. Deskriptive Techniken
- 1.2.3. Elemente und Struktur eines Algorithmus

1.3. Elemente eines Programms
- 1.3.1. Herkunft und Merkmale der Sprache C++
- 1.3.2. Die Entwicklungsumgebung
- 1.3.3. Konzept des Programms
- 1.3.4. Arten von grundlegender Daten
- 1.3.5. Betreiber
- 1.3.6. Ausdrücke
- 1.3.7. Sätze
- 1.3.8. Dateneingabe und -ausgabe

1.4. Kontrollsätze
- 1.4.1. Sätze
- 1.4.2. Verzweigungen
- 1.4.3. Schleifen

1.5. Abstraktion und Modularität: Funktionen
- 1.5.1. Modularer Aufbau
- 1.5.2. Konzept der Funktion und des Nutzens
- 1.5.3. Definition einer Funktion
- 1.5.4. Ausführungsablauf beim Aufruf einer Funktion
- 1.5.5. Prototyp einer Funktion
- 1.5.6. Rückgabe der Ergebnisse
- 1.5.7. Aufrufen einer Funktion: Parameter
- 1.5.8. Übergabe von Parametern per Referenz und per Wert
- 1.5.9. Identifikator des Anwendungsbereichs

1.6. Statische Datenstrukturen
- 1.6.1. *Arrays*
- 1.6.2. Matrizen. Polyeder
- 1.6.3. Suchen und Sortieren
- 1.6.4. Zeichenketten. E/A-Funktionen für Zeichenketten
- 1.6.5. Strukturen. Verbindungen
- 1.6.6. Neue Datentypen

1.7. Dynamische Datenstrukturen: Zeiger
- 1.7.1. Konzept. Definition von Zeiger
- 1.7.2. Operatoren und Operationen mit Zeigern
- 1.7.3. Arrays von Zeigern
- 1.7.4. Zeiger und Arrays
- 1.7.5. Zeiger auf Zeichenketten
- 1.7.6. Zeiger auf Strukturen
- 1.7.7. Multiple Indirektion
- 1.7.8. Zeiger auf Funktionen
- 1.7.9. Übergabe von Funktionen, Strukturen und Arrays als Funktionsparameter

1.8. Dateien
- 1.8.1. Grundlegende Konzepte
- 1.8.2. Dateioperationen
- 1.8.3. Datentypen
- 1.8.4. Organisation von Dateien
- 1.8.5. Einführung in C++ Dateien
- 1.8.6. Handhabung von Dateien

1.9. Rekursion
- 1.9.1. Definition von Rekursion
- 1.9.2. Arten der Rekursion
- 1.9.3. Vorteile und Nachteile
- 1.9.4. Überlegungen
- 1.9.5. Rekursiv-iterative Umwandlung
- 1.9.6. Der Rekursionsstapel

1.10. Prüfung und Dokumentation
- 1.10.1. Programm-Tests
- 1.10.2. *White Box*-Tests
- 1.10.3. *Black Box*-Tests
- 1.10.4. Test-Tools
- 1.10.5. Programm-Dokumentation

Modul 2. Datenstruktur und Algorithmen

2.1. Einführung in Algorithmus-Design-Strategien
- 2.1.1. Rekursion
- 2.1.2. Aufteilen und erobern
- 2.1.3. Andere Strategien

2.2. Effizienz und Analyse von Algorithmen
- 2.2.1. Maßnahmen zur Effizienz
- 2.2.2. Messung der Eingabegröße
- 2.2.3. Messung der Ausführungszeit
- 2.2.4. Schlimmster, bester und durchschnittlicher Fall
- 2.2.5. Asymptotische Notation
- 2.2.6. Kriterien für die mathematische Analyse von nichtrekursiven Algorithmen
- 2.2.7. Mathematische Analyse von rekursiven Algorithmen
- 2.2.8. Empirische Analyse von Algorithmen

2.3. Sortieralgorithmen
- 2.3.1. Konzept der Sortierung
- 2.3.2. Blasen-Sortierung
- 2.3.3. Sortieren durch Auswahl
- 2.3.4. Sortieren durch Einfügen
- 2.3.5. Sortieren nach Zusammenführen (*merge_sort*)
- 2.3.6. Schnelles Sortieren (*quick_sort*)

2.4. Algorithmen mit Bäumen
- 2.4.1. Konzept des Baumes
- 2.4.2. Binäre Bäume
- 2.4.3. Baumpfade
- 2.4.4. Ausdrücke darstellen
- 2.4.5. Geordnete binäre Bäume
- 2.4.6. Ausgeglichene binäre Bäume

2.5. Algorithmen mit *Heaps*
- 2.5.1. *Heaps*
- 2.5.2. Der *Heapsort*-Algorithmus
- 2.5.3. Prioritätswarteschlangen

2.6. Graph-Algorithmen
- 2.6.1. Vertretung
- 2.6.2. Lauf in Breite
- 2.6.3. Lauf in Tiefe
- 2.6.4. Topologische Anordnung

2.7. *Greedy*-Algorithmen
- 2.7.1. Die *Greedy*-Strategie
- 2.7.2. Elemente der *Greedy*-Strategie
- 2.7.3. Währungsumtausch
- 2.7.4. Das Problem des Reisenden
- 2.7.5. Problem mit dem Rucksack

2.8. Minimale Pfadsuche
- 2.8.1. Das Problem des minimalen Pfades
- 2.8.2. Negative Bögen und Zyklen
- 2.8.3. Dijkstra-Algorithmus

2.9. *Greedy*-Algorithmen auf Graphen
- 2.9.1. Der minimal aufspannende Baum
- 2.9.2. Algorithmus von Prim
- 2.9.3. Algorithmus von Kruskal
- 2.9.4. Komplexitätsanalyse

2.10. *Backtracking*
- 2.10.1. Das *Backtracking*
- 2.10.2. Alternative Techniken

Modul 3. Objektorientierte Programmierung

3.1. Einführung in die objektorientierte Programmierung
- 3.1.1. Einführung in die objektorientierte Programmierung
- 3.1.2. Klassen-Design
- 3.1.3. Einführung in UML für die Modellierung von Problemen

3.2. Beziehungen zwischen Klassen
- 3.2.1. Abstraktion und Vererbung
- 3.2.2. Fortgeschrittene Konzepte der Vererbung
- 3.2.3. Polymorphismen
- 3.2.4. Zusammensetzung und Aggregation

3.3. Einführung in *Design Patterns* für objektorientierte Probleme
- 3.3.1. Was sind Entwurfsmuster?
- 3.3.2. *Factory*-Muster
- 3.3.4. *Singleton*-Muster
- 3.3.5. *Observer*-Muster
- 3.3.6. *Composite*-Muster

3.4. Ausnahmen
- 3.4.1. Was sind Ausnahmen?
- 3.4.2. Abfangen und Behandlung von Ausnahmen
- 3.4.3. Werfen von Ausnahmen
- 3.4.4. Erstellung von Ausnahmen

3.5. Benutzeroberflächen
- 3.5.1. Einführung in Qt
- 3.5.2. Positionierung
- 3.5.3. Was sind Ereignisse?
- 3.5.4. Ereignisse: Definition und Erfassung
- 3.5.5. Entwicklung von Benutzeroberflächen

3.6. Einführung in die gleichzeitige Programmierung
- 3.6.1. Einführung in die gleichzeitige Programmierung
- 3.6.2. Der Prozess und das *Thread*-Konzept
- 3.6.3. Interaktion zwischen Prozessen oder *Threads*
- 3.6.4. *Threads* in C++
- 3.6.5. Vor- und Nachteile der gleichzeitigen Programmierung

3.7. *Thread*-Verwaltung und Synchronisierung
- 3.7.1. Lebenszyklus eines *Threads*
- 3.7.2. Die Klasse *Thread*
- 3.7.3. Planung des *Threads*
- 3.7.4. Gruppen von *Threads*
- 3.7.5. Dämonen-*Threads*
- 3.7.6. Synchronisierung
- 3.7.7. Verriegelungsmechanismen
- 3.7.8. Kommunikationsmechanismen
- 3.7.9. Monitore

3.8. Häufige Probleme bei der gleichzeitigen Programmierung
- 3.8.1. Das Erzeuger-Verbraucher-Problem
- 3.8.2. Das Problem von Lesern und Schreibern
- 3.8.3. Das Problem mit dem Abendessen der Philosophen

3.9. Dokumentation und Prüfung von Software
- 3.9.1. Warum ist es wichtig, Software zu dokumentieren?
- 3.9.2. Design-Dokumentation
- 3.9.3. Verwendung von Tools zur Dokumentation

3.10. Software-Tests
- 3.10.1. Einführung in die Softwareprüfung
- 3.10.2. Arten von Tests
- 3.10.3. Einheitstest
- 3.10.4. Integrationstests
- 3.10.5. Validierungstest
- 3.10.6. Systemprüfung

Modul 4. Spielkonsolen und Geräte

4.1. Geschichte der Videospielprogrammierung
- 4.1.1. Atari (1977-1985)
- 4.1.2. NES und SNES (1985-1995)
- 4.1.3. PlayStation/PlayStation 2 (1995-2005)
- 4.1.4. Xbox 360, PS3 und Wii (2005-2013)
- 4.1.5. Xbox One, PS4 und Wii U-Switch (2013-heute)
- 4.1.6. Die Zukunft

4.2. Geschichte des Gameplays in Videospielen
- 4.2.1. Einführung
- 4.2.2. Sozialer Kontext
- 4.2.3. Strukturelles Diagramm
- 4.2.4. Zukunft

4.3. Anpassung an die moderne Zeit
- 4.3.1. Bewegungsbasierte Spiele
- 4.3.2. *Virtual Reality*
- 4.3.3. *Augmented Reality*
- 4.3.4. Gemischte Realität

4.4. *Unity: Scripting* I und Beispiele
- 4.4.1. Was ist ein *Script*?
- 4.4.2. Unser erstes *Script*
- 4.4.3. Hinzufügen eines *Scripts*
- 4.4.4. Öffnen eines *Scripts*
- 4.4.5. *MonoBehaviour*
- 4.4.6. *Debugging*

4.5. *Unity: Scripting* II und Beispiele
- 4.5.1. Tastatur- und Mauseingabe
- 4.5.2. Raycast
- 4.5.3. Instanziierung
- 4.5.4. Variablen
- 4.5.5. Öffentliche und serialisierte Variablen

4.6. *Unity: Scripting* III und Beispiele
- 4.6.1. Beschaffung von Komponenten
- 4.6.2. Komponenten modifizieren
- 4.6.3. Testen
- 4.6.4. Mehrere Objekte
- 4.6.5. *Colliders* und *Triggers*
- 4.6.6. Quaternionen

4.7. Peripheriegeräte
- 4.7.1. Entwicklung und Klassifizierung
- 4.7.2. Peripheriegeräte und Schnittstellen
- 4.7.3. Aktuelle Peripheriegeräte
- 4.7.4. Nahe Zukunft

4.8. Videospiele: Zukunftsperspektiven
- 4.8.1. Cloud-basiertes Spielen
- 4.8.2. Abwesenheit von Controllern
- 4.8.3. Immersive Realität
- 4.8.4. Andere Alternativen

4.9. Architektur
- 4.9.1. Besondere Anforderungen für Videospiele
- 4.9.2. Entwicklung der Architektur
- 4.9.3. Zeitgenössische Architektur
- 4.9.4. Unterschiede zwischen den Architekturen

4.10. Entwicklungskits und ihre Evolution
- 4.10.1. Einführung
- 4.10.2. Entwicklungskits der dritten Generation
- 4.10.3. Entwicklungskits der vierten Generation
- 4.10.4. Entwicklungskits der fünften Generation
- 4.10.5. Entwicklungskits der sechsten Generation

Modul 5. Softwaretechnik

5.1. Einführung in die Softwaretechnik und Modellierung
- 5.1.1. Die Natur der Software
- 5.1.2. Die Besonderheit von Webapps
- 5.1.3. Softwaretechnik
- 5.1.4. Der Software-Prozess
- 5.1.5. Die Praxis der Softwaretechnik
- 5.1.6. Software-Mythen
- 5.1.7. Wie alles beginnt
- 5.1.8. Objektorientierte Konzepte
- 5.1.9. Einführung in UML

5.2. Der Software-Prozess
- 5.2.1. Ein allgemeines Prozessmodell
- 5.2.2. Vorgeschriebene Prozessmodelle
- 5.2.3. Spezialisierte Prozessmodelle
- 5.2.4. Einheitlicher Prozess
- 5.2.5. Personal- und Teamprozessmodelle
- 5.2.6. Was ist Agilität?
- 5.2.7. Was ist ein agiler Prozess?
- 5.2.8. Scrum
- 5.2.9. Werkzeugkasten für agile Prozesse

5.3. Prinzipien als Leitfaden für die Praxis der Softwareentwicklung
- 5.3.1. Prinzipien als Leitfaden des Prozesses
- 5.3.2. Prinzipien als Leitfaden für die Praxis
- 5.3.3. Prinzipien der Kommunikation
- 5.3.4. Prinzipien der Planung
- 5.3.5. Prinzipien der Modellierung
- 5.3.6. Prinzipien der Konstruktion
- 5.3.7. Prinzipien der Einführung

5.4. Verständnis der Anforderungen
- 5.4.1. Anforderungsmanagement
- 5.4.2. Schaffung der Grundlagen
- 5.4.3. Bedarfsermittlung
- 5.4.4. Entwicklung von Anwendungsfällen
- 5.4.5. Ausarbeitung des Anforderungsmodells
- 5.4.6. Aushandeln von Anforderungen
- 5.4.7. Validierung der Anforderungen

5.5. Modellierung der Anforderungen: Szenarien, Informationen und Arten der Analyse
- 5.5.1. Analyse der Anforderungen
- 5.5.2. Szenario-basiertes Modell
- 5.5.3. UML-Modelle, die den Anwendungsfall liefern
- 5.5.4. Konzepte der Datenmodellierung
- 5.5.5. Klassen-basiertes Modell
- 5.5.6. Klassendiagramme

5.6. Modellierung der Anforderungen: Fluss, Verhalten und Muster
- 5.6.1. Anforderungen die die Strategien gestalten
- 5.6.2. Flussorientierte Modellierung
- 5.6.3. Zustandsdiagramme
- 5.6.4. Erstellung eines Verhaltensmodells
- 5.6.5. Sequenzdiagramme
- 5.6.6. Kommunikationsdiagramme
- 5.6.7. Muster für die Modellierung von Anforderungen

5.7. Konzepte des Designs
- 5.7.1. Design im Kontext der Softwaretechnik
- 5.7.2. Der Entwurfsprozess
- 5.7.3. Konzepte des Designs
- 5.7.4. Objektorientierte Konzepte des Designs
- 5.7.5. Das Designmodell

5.8. Design der Architektur
- 5.8.1. Software-Architektur
- 5.8.2. Architektonische Gattungen
- 5.8.3. Architektonische Stile
- 5.8.4. Architektonisches Design
- 5.8.5. Entwicklung von alternativen Designs für die Architektur
- 5.8.6. Mapping der Architektur mit Hilfe von Datenflüssen

5.9. Design auf Komponentenebene und musterbasierter Entwurf
- 5.9.1. Was ist eine Komponente?
- 5.9.2. Klassenbasiertes Komponentendesign
- 5.9.3. Verwirklichung des Designs auf Komponentenebene
- 5.9.4. Design der traditionellen Komponenten
- 5.9.5. Komponentenbasierte Entwicklung
- 5.9.6. Entwurfsmuster
- 5.9.7. Musterbasiertes Softwaredesign
- 5.9.8. Architektonische Muster
- 5.9.9. Musterdesign auf Komponentenebene
- 5.9.10. Musterdesign für Benutzeroberflächen

5.10. Softwarequalität und Projektmanagement
- 5.10.1. Qualität
- 5.10.2. Software-Qualität
- 5.10.3. Das Dilemma der Softwarequalität
- 5.10.4. Erreichen von Softwarequalität
- 5.10.5. Software-Qualitätssicherung
- 5.10.6. Das administrative Spektrum
- 5.10.7. Das Personal
- 5.10.8. Das Produkt
- 5.10.9. Der Prozess
- 5.10.10. Das Projekt
- 5.10.11. Grundsätze und Praktiken

Modul 6. Videospiel-Engines

6.1. Videospiele und IKTs
- 6.1.1. Einführung
- 6.1.2. Gelegenheiten
- 6.1.3. Herausforderungen
- 6.1.4. Schlussfolgerungen

6.2. Geschichte der Videospiel-Engines
- 6.2.1. Einführung
- 6.2.2. Atari-Ära
- 6.2.3. 1980er-Ära
- 6.2.4. Erste Engines. 90er-Ära
- 6.2.5. Aktuelle Motoren

6.3. Videospiel-Engines
- 6.3.1. Typen von Engines
- 6.3.2. Teile einer Videospiel-Engine
- 6.3.3. Aktuelle Motoren
- 6.3.4. Auswahl eines Motors für unser Projekt

6.4. *Motor Game Maker*
- 6.4.1. Einführung
- 6.4.2. Entwurf eines Szenarios
- 6.4.3. *Sprites* und Animationen
- 6.4.4. Kollisionen
- 6.4.5. *Scripting* in GML

6.5. Motor Unreal Engine 4: Einführung
- 6.5.1. Was ist die Unreal Engine 4? Was ist ihre Philosophie?
- 6.5.2. Materialien
- 6.5.3. UI
- 6.5.4. Animationen
- 6.5.5. Partikel System
- 6.5.6. Künstliche Intelligenz
- 6.5.7. FPS

6.6. Motor Unreal Engine 4: *Visual Scripting*
- 6.6.1. *Blueprint*-Philosophie und *Visual Scripting*
- 6.6.2. *Debugging*
- 6.6.3. Arten von Variablen
- 6.6.4. Grundlegende Flusskontrolle

6.7. Motor Unity 5
- 6.7.1. Programmieren in C# und Visual Studio
- 6.7.2. Erschaffen von Prefabs
- 6.7.3. Verwendung von *Gizmos* zur Steuerung von Videospielen
- 6.7.4. Adaptiver Motor: 2D und 3D

6.8. Godot-Motor
- 6.8.1. Godot Design-Philosophie
- 6.8.2. Objektorientiertes Design und Komposition
- 6.8.3. *All-in-One*-Paket
- 6.8.4. Freie und von der Gemeinschaft betriebene Software

6.9. RPG Maker-Engine
- 6.9.1. RPG Maker-Philosophie
- 6.9.2. Als Bezug nehmen
- 6.9.3. Ein Spiel mit Persönlichkeit schaffen
- 6.9.4. Erfolgreiche kommerzielle Spiele

6.10. Motor Source 2
- 6.10.1. Source 2-Philosophie
- 6.10.2. Source und Source 2: Entwicklung
- 6.10.3. Verwendung der Gemeinschaften: Audiovisuelle Inhalte und Videospiele
- 6.10.4. Die Zukunft der Source 2 Engine
- 6.10.5. Mods und erfolgreiche Spiele

Modul 7. Intelligente Systeme

7.1. Agententheorie
- 7.1.1. Geschichte des Konzepts
- 7.1.2. Definition von Agent
- 7.1.3. Agenten in der künstlichen Intelligenz
- 7.1.4. Agenten in der Softwareentwicklung

7.2. Agenten-Architekturen
- 7.2.1. Der Denkprozess eines Agenten
- 7.2.2. Reaktive Wirkstoffe
- 7.2.3. Deduktive Agenten
- 7.2.4. Hybride Agenten
- 7.2.5. Vergleich

7.3. Informationen und Wissen
- 7.3.1. Unterscheidung zwischen Daten, Informationen und Wissen
- 7.3.2. Bewertung der Datenqualität
- 7.3.3. Methoden der Datenerfassung
- 7.3.4. Methoden der Informationsbeschaffung
- 7.3.5. Methoden zum Wissenserwerb

7.4. Darstellung von Wissen
- 7.4.1. Die Bedeutung der Wissensdarstellung
- 7.4.2. Definition der Wissensrepräsentation durch ihre Rollen
- 7.4.3. Merkmale einer Wissensrepräsentation

7.5. Ontologien
- 7.5.1. Einführung in Metadaten
- 7.5.2. Philosophisches Konzept der Ontologie
- 7.5.3. Computergestütztes Konzept der Ontologie
- 7.5.4. Bereichsontologien und Ontologien auf höherer Ebene
- 7.5.5. Wie man eine Ontologie erstellt

7.6. Ontologiesprachen und Software für die Erstellung von Ontologien
- 7.6.1. RDF-Tripel, Turtle und N3
- 7.6.2. RDF-Schema
- 7.6.3. OWL
- 7.6.4. SPARQL
- 7.6.5. Einführung in die verschiedenen Tools für die Erstellung von Ontologien
- 7.6.6. Installation und Verwendung von Protégé

7.7. Das semantische Web
- 7.7.1. Der aktuelle Stand und die Zukunft des semantischen Webs
- 7.7.2. Anwendungen des semantischen Webs

7.8. Andere Modelle der Wissensdarstellung
- 7.8.1. Wortschatz
- 7.8.2. Globale Sicht
- 7.8.3. Taxonomie
- 7.8.4. Thesauri
- 7.8.5. Folksonomien
- 7.8.6. Vergleich
- 7.8.7. *Mind Maps*

7.9. Bewertung und Integration von Wissensrepräsentationen
- 7.9.1. Logik nullter Ordnung
- 7.9.2. Logik erster Ordnung
- 7.9.3. Beschreibende Logik
- 7.9.4. Beziehung zwischen verschiedenen Arten von Logik
- 7.9.5. Prolog: Programmierung auf Basis der Logik erster Ordnung

7.10. Semantische *Reasoner*, wissensbasierte Systeme und Expertensysteme
- 7.10.1. Konzept des *Reasoners*
- 7.10.2. Anwendungen eines *Reasoners*
- 7.10.3. Wissensbasierte Systeme
- 7.10.4. MYCIN, Geschichte der Expertensysteme
- 7.10.5. Elemente und Architektur von Expertensystemen
- 7.10.6. Erstellung von Expertensystemen

Modul 8. Echtzeit-Programmierung

8.1. Grundlegende Konzepte der parallelen Programmierung
- 8.1.1. Grundlegende Konzepte
- 8.1.2. Parallelität
- 8.1.3. Vorteile der Parallelität
- 8.1.4. Parallelität und Hardware

8.2. Grundlegende Strukturen zur Unterstützung der Parallelität in Java
- 8.2.1. Parallelität in Java
- 8.2.2. *Threads* erstellen
- 8.2.3. Methoden
- 8.2.4. Synchronisierung

8.3. *Threads*, Lebenszyklus, Prioritäten, Unterbrechungen, Zustände, Executors
- 8.3.1. *Threads*
- 8.3.2. Lebenszyklus
- 8.3.3. Prioritäten
- 8.3.4. Unterbrechungen
- 8.3.5. Staaten
- 8.3.6. Umsetzer

8.4. Gegenseitiger Ausschluss
- 8.4.1. Was bedeutet gegenseitiger Ausschluss?
- 8.4.2. Dekkers Algorithmus
- 8.4.3. Petersons Algorithmus
- 8.4.4. Gegenseitiger Ausschluss in Java

8.5. Abhängigkeiten vom Zustand
- 8.5.1. Injektion von Abhängigkeiten
- 8.5.2. Java-Implementierung des Musters
- 8.5.3. Wege zur Injektion von Abhängigkeiten
- 8.5.4. Beispiel

8.6. Entwurfsmuster
- 8.6.1. Einführung
- 8.6.2. Erzeugungsmuster
- 8.6.3. Struktur-Muster
- 8.6.4. Verhaltensmuster

8.7. Verwendung von Java-Bibliotheken
- 8.7.1. Was sind Bibliotheken in Java?
- 8.7.2. Mockito-All, Mockito-Core
- 8.7.3. Guava
- 8.7.4. Commons-io
- 8.7.5. Commons-lang, Commons-lang3

8.8. Shaders-Programmierung
- 8.8.1. 3D-Pipeline und Raster
- 8.8.2. Vertex Shading
- 8.8.3. Pixel Shading: Beleuchtung I
- 8.8.4. Pixel Shading: Beleuchtung II
- 8.8.5. Post-Effekte

8.9. Programmierung in Echtzeit
- 8.9.1. Einführung
- 8.9.2. Verarbeitung von Unterbrechungen
- 8.9.3. Synchronisierung und Kommunikation zwischen Prozessen
- 8.9.4. Planungssysteme in Echtzeit

8.10. Planung in Echtzeit
- 8.10.1. Konzepte
- 8.10.2. Referenzmodell für Echtzeitsysteme
- 8.10.3. Planungspolitik
- 8.10.4. Zyklische Planer
- 8.10.5. Planer mit statischen Eigenschaften
- 8.10.6. Planer mit dynamischen Eigenschaften

Modul 9. Design und Entwicklung von Webspielen

9.1. Ursprünge und Standards des Webs
- 9.1.1. Die Ursprünge des Internets
- 9.1.2. Die Entstehung des World Wide Web
- 9.1.3. Aufkommen von Webstandards
- 9.1.4. Der Aufstieg der Webstandards

9.2. HTTP und Client-Server-Struktur
- 9.2.1. Client-Server-Rolle
- 9.2.2. Client-Server-Kommunikation
- 9.2.3. Jüngste Geschichte
- 9.2.4. Zentralisierte Datenverarbeitung

9.3. Web-Programmierung: Einführung
- 9.3.1. Grundlegende Konzepte
- 9.3.2. Einrichten eines Webservers
- 9.3.3. HTML5-Grundlagen
- 9.3.4. HTML-Formulare

9.4. Einführung in HTML und Beispiele
- 9.4.1. Geschichte von HTML5
- 9.4.2. Elemente von HTML5
- 9.4.3. APIS
- 9.4.4. CCS3

9.5. *Document Object Model*
- 9.5.1. Was ist das *Document Object Model*?
- 9.5.2. Verwendung von DOCTYPE
- 9.5.3. Die Bedeutung der Validierung von HTML
- 9.5.4. Zugriff auf Elemente
- 9.5.5. Elemente und Text erstellen
- 9.5.6. InnerHTML verwenden
- 9.5.7. Ein Textelement oder einen Knoten löschen
- 9.5.8. Lesen und Schreiben der Attribute eines Elements
- 9.5.9. Manipulation von Elementstilen
- 9.5.10. Mehrere Dateien auf einmal anhängen

9.6. Einführung in CSS und Beispiele
- 9.6.1. CSS3-Syntax
- 9.6.2. Stil-Blätter
- 9.6.3. Tags
- 9.6.4. Selektoren
- 9.6.5. Webgestaltung mit CSS

9.7. Einführung in JavaScript und Beispiele
- 9.7.1. Was ist JavaScript?
- 9.7.2. Kurze Geschichte der Sprache
- 9.7.3. JavaScript-Versionen
- 9.7.4. Ein Dialogfeld anzeigen
- 9.7.5. JavaScript-Syntax
- 9.7.6. *Scripts* verstehen
- 9.7.7. Räume
- 9.7.8. Kommentare
- 9.7.9. Funktionen
- 9.7.10. Seiteninternes und externes JavaScript

9.8. Funktionen in JavaScript
- 9.8.1. Funktionsdeklarationen
- 9.8.2. Funktion Ausdrücke
- 9.8.3. Funktionen aufrufen
- 9.8.4. Rekursion
- 9.8.5. Verschachtelte Funktionen und Schließungen

9.8.6. Variable Konservierung
9.8.7. Mehrfach verschachtelte Funktionen
9.8.8. Namenskonflikte
9.8.9. Schließungen
9.8.10. Parameter einer Funktion
9.9. PlayCanvas für die Entwicklung von Webspielen
9.9.1. Was ist PlayCanvas?
9.9.2. Projekt-Konfiguration
9.9.3. Ein Objekt erstellen
9.9.4. Hinzufügen von Physiken
9.9.5. Hinzufügen eines Modells
9.9.6. Ändern der Schwerkraft- und Szeneneinstellungen
9.9.7. *Scripts* ausführen
9.9.8. Kamera-Steuerungen
9.10. *Phaser* für die Entwicklung von Webspielen
9.10.1. Was ist *Phaser*?
9.10.2. Ressourcen laden
9.10.3. Die Welt bauen
9.10.4. Die Plattformen
9.10.5. Der Spieler
9.10.6. Hinzufügen von Physiken
9.10.7. Verwendung der Tastatur
9.10.8. Aufnehmen von *Pickups*
9.10.9. Punkte und Wertung
9.10.10. *Bounce*-Pumpen

Modul 10. Multiplayer-Netzwerke und -Systeme

10.1. Geschichte und Entwicklung von Multiplayer-Spielen
10.1.1. Das Jahrzehnt 1970: Erste Multiplayer-Spiele
10.1.2. 1990er Jahre: Duke Nukem, Doom, Quake
10.1.3. Der Aufstieg der Multiplayer-Videospiele
10.1.4. Lokaler und Online-Multiplayer
10.1.5. Partyspiele
10.2. Multiplayer-Geschäftsmodelle
10.2.1. Entstehung und Funktionsweise von neuen Geschäftsmodellen
10.2.2. Online-Verkaufsdienstleistungen
10.2.3. Frei zum Spielen
10.2.4. Micropayments
10.2.5. Werbung
10.2.6. Abonnement mit monatlichen Zahlungen
10.2.7. *Pay-per-play*
10.2.8. Testen vor dem Kauf
10.3. Lokale Spiele und vernetzte Spiele
10.3.1. Lokale Spiele: Erste Schritte
10.3.2. Partyspiele: Nintendo und Familienzusammengehörigkeit
10.3.3. Netzwerkspiele: Anfänge
10.3.4. Entwicklung von Netzwerkspielen
10.4. OSI-Modell: Schichten I
10.4.1. OSI-Modell: Einleitung
10.4.2. Physikalische Schicht
10.4.3. Datenübertragungsschicht
10.4.4. Netzwerkschicht
10.5. OSI-Modell: Schichten II
10.5.1. Transportschicht
10.5.2. Sitzungsschicht
10.5.3. Präsentationsschicht
10.5.4. Anwendungsschicht

10.6. Computernetzwerke und das Internet
- 10.6.1. Was ist ein Computernetzwerk?
- 10.6.2. Software
- 10.6.3. Hardware
- 10.6.4. Server
- 10.6.5. Netzwerkspeicher
- 10.6.6. Netzwerk-Protokolle

10.7. Mobile und drahtlose Netzwerke
- 10.7.1. Mobiles Netzwerk
- 10.7.2. Drahtloses Netzwerk
- 10.7.3. Betrieb von mobilen Netzwerken
- 10.7.4. Digitale Technologie

10.8. Sicherheit
- 10.8.1. Persönliche Sicherheit
- 10.8.2. *Hacks* und *Cheats* in Videospielen
- 10.8.3. Anti-Betrugsmaßnahmen-Sicherheit
- 10.8.4. Analyse von Sicherheitssystemen gegen Betrug

10.9. Multiplayer-Systeme: Server
- 10.9.1. *Server-Hosting*
- 10.9.2. MMO-Videospiele
- 10.9.3. Dedizierte Videospiel-Server
- 10.9.4. LAN *Parties*

10.10. Design und Programmierung von Multiplayer-Videospielen
- 10.10.1. Grundlagen der Entwicklung von Multiplayer-Spielen in Unreal
- 10.10.2. Grundlagen der Entwicklung von Multiplayer-Spielen in Unity
- 10.10.3. Wie gestaltet man ein Multiplayer-Spiel unterhaltsam?
- 10.10.4. Jenseits eines Controllers: Innovation in der Multiplayer-Steuerung

Wenn Sie eine großartige Karriere bei der Programmierung weltberühmter Videospiele anstreben, ist dies der richtige Abschluss für Sie"

05

Methodik

Dieses Fortbildungsprogramm bietet eine andere Art des Lernens. Unsere Methodik wird durch eine zyklische Lernmethode entwickelt: ***das Relearning***.
Dieses Lehrsystem wird z. B. an den renommiertesten medizinischen Fakultäten der Welt angewandt und wird von wichtigen Publikationen wie dem ***New England Journal of Medicine*** als eines der effektivsten angesehen.

Entdecken Sie Relearning, ein System, das das herkömmliche lineare Lernen hinter sich lässt und Sie durch zyklische Lehrsysteme führt: eine Art des Lernens, die sich als äußerst effektiv erwiesen hat, insbesondere in Fächern, die Auswendiglernen erfordern"

Fallstudie zur Kontextualisierung aller Inhalte

Unser Programm bietet eine revolutionäre Methode zur Entwicklung von Fähigkeiten und Kenntnissen. Unser Ziel ist es, Kompetenzen in einem sich wandelnden, wettbewerbsorientierten und sehr anspruchsvollen Umfeld zu stärken.

Mit TECH werden Sie eine Art des Lernens erleben, die an den Grundlagen der traditionellen Universitäten auf der ganzen Welt rüttelt"

Sie werden Zugang zu einem Lernsystem haben, das auf Wiederholung basiert, mit natürlichem und progressivem Unterricht während des gesamten Lehrplans.

Der Student wird durch gemeinschaftliche Aktivitäten und reale Fälle lernen, wie man komplexe Situationen in realen Geschäftsumgebungen löst.

Eine innovative und andersartige Lernmethode

Dieses TECH-Programm ist ein von Grund auf neu entwickeltes, intensives Lehrprogramm, das die anspruchsvollsten Herausforderungen und Entscheidungen in diesem Bereich sowohl auf nationaler als auch auf internationaler Ebene vorsieht. Dank dieser Methodik wird das persönliche und berufliche Wachstum gefördert und ein entscheidender Schritt in Richtung Erfolg gemacht. Die Fallmethode, die Technik, die diesem Inhalt zugrunde liegt, gewährleistet, dass die aktuellste wirtschaftliche, soziale und berufliche Realität berücksichtigt wird.

> *Unser Programm bereitet Sie darauf vor, sich neuen Herausforderungen in einem unsicheren Umfeld zu stellen und in Ihrer Karriere erfolgreich zu sein"*

Die Fallmethode ist das am weitesten verbreitete Lernsystem an den besten Informatikschulen der Welt, seit es sie gibt. Die Fallmethode wurde 1912 entwickelt, damit Jurastudenten das Recht nicht nur auf der Grundlage theoretischer Inhalte erlernen. Sie bestand darin, ihnen reale komplexe Situationen zu präsentieren, damit sie fundierte Entscheidungen treffen und Werturteile darüber fällen konnten, wie diese zu lösen sind. Sie wurde 1924 als Standardlehrmethode in Harvard etabliert.

Was sollte eine Fachkraft in einer bestimmten Situation tun? Mit dieser Frage konfrontieren wir Sie in der Fallmethode, einer handlungsorientierten Lernmethode. Während des gesamten Kurses werden die Studenten mit mehreren realen Fällen konfrontiert. Sie müssen ihr gesamtes Wissen integrieren, recherchieren, argumentieren und ihre Ideen und Entscheidungen verteidigen.

Relearning Methodology

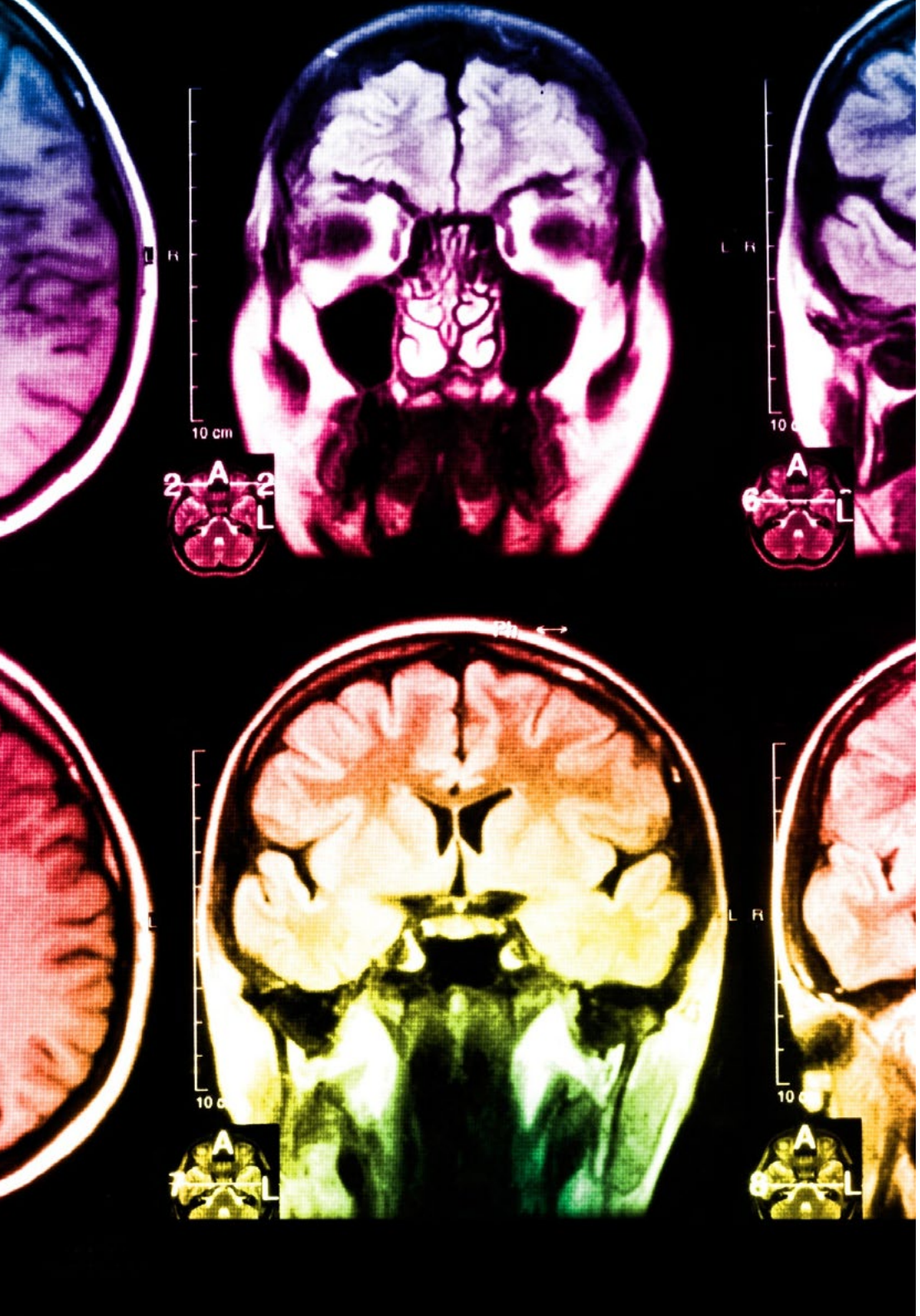
TECH kombiniert die Methodik der Fallstudien effektiv mit einem 100%igen Online-Lernsystem, das auf Wiederholung basiert und in jeder Lektion verschiedene didaktische Elemente kombiniert.

Wir ergänzen die Fallstudie mit der besten 100%igen Online-Lehrmethode: Relearning.

Im Jahr 2019 erzielten wir die besten Lernergebnisse aller spanischsprachigen Online-Universitäten der Welt.

Bei TECH lernen Sie mit einer hochmodernen Methodik, die darauf ausgerichtet ist, die Führungskräfte der Zukunft zu spezialisieren. Diese Methode, die an der Spitze der weltweiten Pädagogik steht, wird Relearning genannt.

Unsere Universität ist die einzige in der spanischsprachigen Welt, die für die Anwendung dieser erfolgreichen Methode zugelassen ist. Im Jahr 2019 ist es uns gelungen, die Gesamtzufriedenheit unserer Studenten (Qualität der Lehre, Qualität der Materialien, Kursstruktur, Ziele...) in Bezug auf die Indikatoren der besten spanischsprachigen Online-Universität zu verbessern.



In unserem Programm ist das Lernen kein linearer Prozess, sondern erfolgt in einer Spirale (lernen, verlernen, vergessen und neu lernen). Daher wird jedes dieser Elemente konzentrisch kombiniert. Mit dieser Methode wurden mehr als 650.000 Hochschulabsolventen mit beispiellosem Erfolg in so unterschiedlichen Bereichen wie Biochemie, Genetik, Chirurgie, internationales Recht, Managementfähigkeiten, Sportwissenschaft, Philosophie, Recht, Ingenieurwesen, Journalismus, Geschichte, Finanzmärkte und -instrumente fortgebildet. Dies alles in einem sehr anspruchsvollen Umfeld mit einer Studentenschaft mit hohem sozioökonomischem Profil und einem Durchschnittsalter von 43,5 Jahren.

Das Relearning ermöglicht es Ihnen, mit weniger Aufwand und mehr Leistung zu lernen, sich mehr auf Ihre Spezialisierung einzulassen, einen kritischen Geist zu entwickeln, Argumente zu verteidigen und Meinungen zu kontrastieren: eine direkte Gleichung zum Erfolg.

Nach den neuesten wissenschaftlichen Erkenntnissen der Neurowissenschaften wissen wir nicht nur, wie wir Informationen, Ideen, Bilder und Erinnerungen organisieren, sondern auch, dass der Ort und der Kontext, in dem wir etwas gelernt haben, von grundlegender Bedeutung dafür sind, dass wir uns daran erinnern und es im Hippocampus speichern können, um es in unserem Langzeitgedächtnis zu behalten.

Auf diese Weise sind die verschiedenen Elemente unseres Programms im Rahmen des so genannten Neurocognitive Context-Dependent E-Learning mit dem Kontext verbunden, in dem der Teilnehmer seine berufliche Praxis entwickelt.

Dieses Programm bietet die besten Lehrmaterialien, die sorgfältig für Fachleute aufbereitet sind:

Studienmaterial

Alle didaktischen Inhalte werden von den Fachleuten, die den Kurs unterrichten werden, speziell für den Kurs erstellt, so dass die didaktische Entwicklung wirklich spezifisch und konkret ist.

Diese Inhalte werden dann auf das audiovisuelle Format angewendet, um die Online-Arbeitsmethode von TECH zu schaffen. All dies mit den neuesten Techniken, die in jedem einzelnen der Materialien, die dem Studenten zur Verfügung gestellt werden, qualitativ hochwertige Elemente bieten.

Meisterklassen

Die Nützlichkeit der Expertenbeobachtung ist wissenschaftlich belegt.

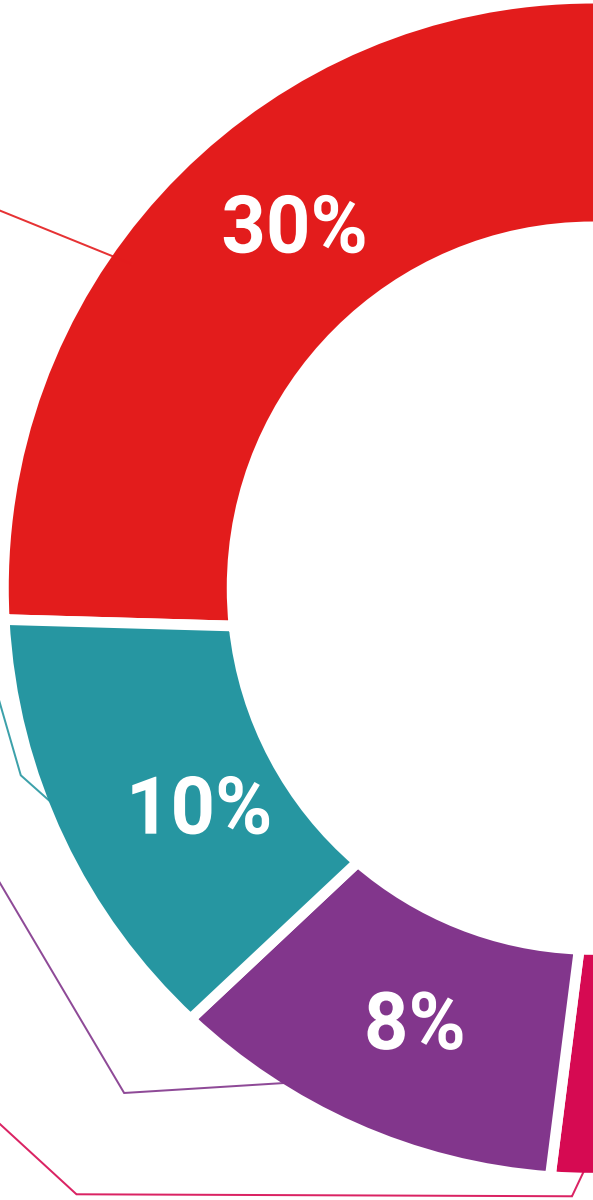
Das sogenannte Learning from an Expert festigt das Wissen und das Gedächtnis und schafft Vertrauen für zukünftige schwierige Entscheidungen.

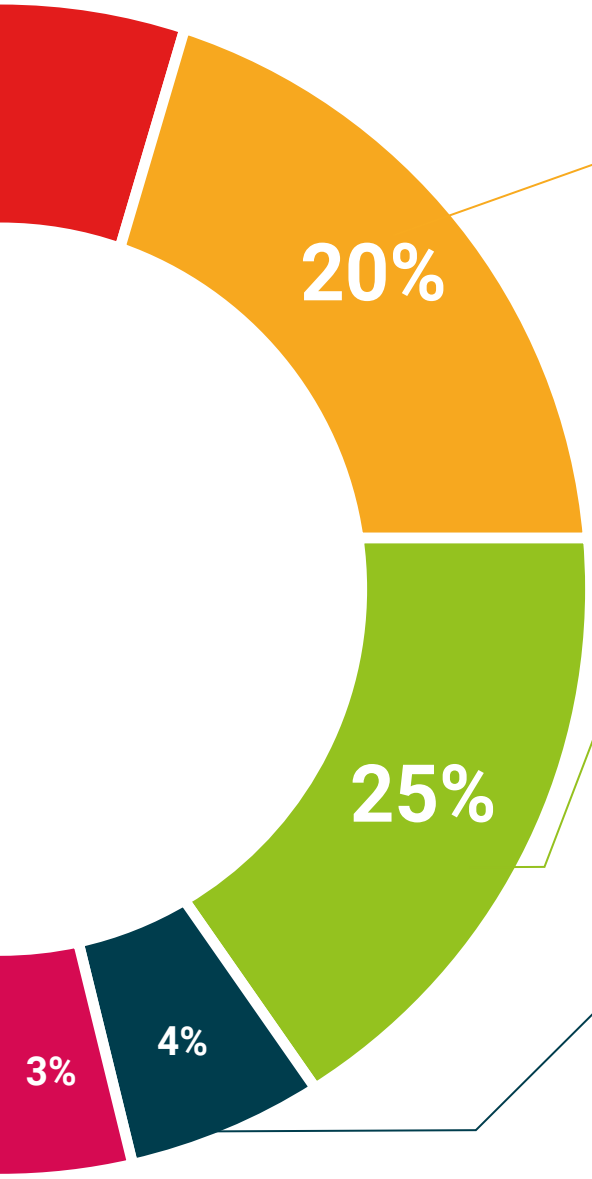
Übungen für Fertigkeiten und Kompetenzen

Sie werden Aktivitäten durchführen, um spezifische Kompetenzen und Fertigkeiten in jedem Fachbereich zu entwickeln. Übungen und Aktivitäten zum Erwerb und zur Entwicklung der Fähigkeiten und Fertigkeiten, die ein Spezialist im Rahmen der Globalisierung, in der wir leben, entwickeln muss.

Weitere Lektüren

Aktuelle Artikel, Konsensdokumente und internationale Leitfäden, u. a. In der virtuellen Bibliothek von TECH hat der Student Zugang zu allem, was er für seine Fortbildung benötigt.





Case Studies

Sie werden eine Auswahl der besten Fallstudien vervollständigen, die speziell für diese Qualifizierung ausgewählt wurden. Die Fälle werden von den besten Spezialisten der internationalen Szene präsentiert, analysiert und betreut.

Interaktive Zusammenfassungen

Das TECH-Team präsentiert die Inhalte auf attraktive und dynamische Weise in multimedialen Pillen, die Audios, Videos, Bilder, Diagramme und konzeptionelle Karten enthalten, um das Wissen zu vertiefen.

Dieses einzigartige Bildungssystem für die Präsentation multimedialer Inhalte wurde von Microsoft als "Europäische Erfolgsgeschichte" ausgezeichnet.

Testing & Retesting

Die Kenntnisse des Studenten werden während des gesamten Programms regelmäßig durch Bewertungs- und Selbsteinschätzungsaktivitäten und -übungen beurteilt und neu bewertet, so dass der Student überprüfen kann, wie er seine Ziele erreicht.

06

Qualifizierung

Der Privater Masterstudiengang in Programmierung von Videospielen garantiert neben der präzisesten und aktuellsten Fortbildung auch den Zugang zu einem von der TECH Technologischen Universität ausgestellten Diplom.

Schließen Sie dieses Programm erfolgreich ab und erhalten Sie Ihren Universitätsabschluss ohne lästige Reisen oder Formalitäten”

Dieser **Privater Masterstudiengang in Programmierung von Videospielen** enthält das vollständigste und aktuellste Programm auf dem Markt.

Sobald der Student die Prüfungen bestanden hat, erhält er/sie per Post* mit Empfangsbestätigung das entsprechende Diplom, ausgestellt von der **TECH Technologischen Universität.**

Das von **TECH Technologische Universität** ausgestellte Diplom drückt die erworbene Qualifikation aus und entspricht den Anforderungen, die in der Regel von Stellenbörsen, Auswahlprüfungen und Berufsbildungsausschüssen verlangt werden.

Titel: **Privater Masterstudiengang in Programmierung von Videospielen**

Anzahl der offiziellen Arbeitsstunden: **1.500 Std.**


tech technologische universität

Verleiht dieses

DIPLOM

an

Herr/Frau _____, mit Ausweis-Nr. _____

Für den erfolgreichen Abschluss und die Akkreditierung des Programms

PRIVATER MASTERSTUDIENGANG

in

Programmierung von Videospielen

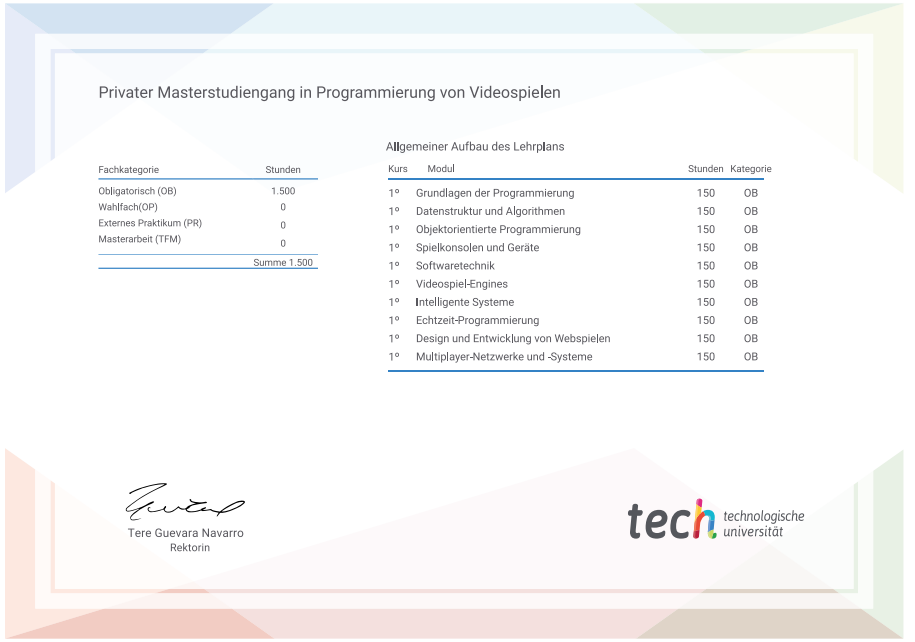
Es handelt sich um einen von dieser Universität verliehenen Abschluss, mit einer Dauer von 1.500 Stunden, mit Anfangsdatum tt/mm/jjjj und Enddatum tt/mm/jjjj.

TECH ist eine private Hochschuleinrichtung, die seit dem 28. Juni 2018 vom Ministerium für öffentliche Bildung anerkannt ist.

Zum 17. Juni 2020

Tere Guevara Navarro
Rektorin

einzigartiger Code TECH: AFWOR23S techtitute.com/titel


Privater Masterstudiengang in Programmierung von Videospielen

Fachkategorie	Stunden
Obligatorisch (OB)	1.500
Wahlfach(OP)	0
Externes Praktikum (PR)	0
Masterarbeit (TFM)	0
	Summe 1.500

Allgemeiner Aufbau des Lehrplans

Kurs	Modul	Stunden	Kategorie
1°	Grundlagen der Programmierung	150	OB
1°	Datenstruktur und Algorithmen	150	OB
1°	Objektorientierte Programmierung	150	OB
1°	Spielkonsolen und Geräte	150	OB
1°	Softwaretechnik	150	OB
1°	Videospiel-Engines	150	OB
1°	Intelligente Systeme	150	OB
1°	Echtzeit-Programmierung	150	OB
1°	Design und Entwicklung von Webspielen	150	OB
1°	Multiplayer-Netzwerke und -Systeme	150	OB

Tere Guevara Navarro
Rektorin

tech technologische universität

*Haager Apostille. Für den Fall, dass der Student die Haager Apostille für sein Papierdiplom beantragt, wird TECH EDUCATION die notwendigen Vorkehrungen treffen, um diese gegen eine zusätzliche Gebühr zu beschaffen.

tech technologische universität

Privater Masterstudiengang Programmierung von Videospielen

- Modalität: **online**
- Dauer: **12 Monate**
- Qualifizierung: **TECH Technologische Universität**
- Aufwand: **16 Std./Woche**
- Zeitplan: **in Ihrem eigenen Tempo**
- Prüfungen: **online**

Privater Masterstudiengang
Programmierung von Videospielen

tech technologische universität